Yisroel Arnson

CMSC 330 7384

12/10/2023

Project #2

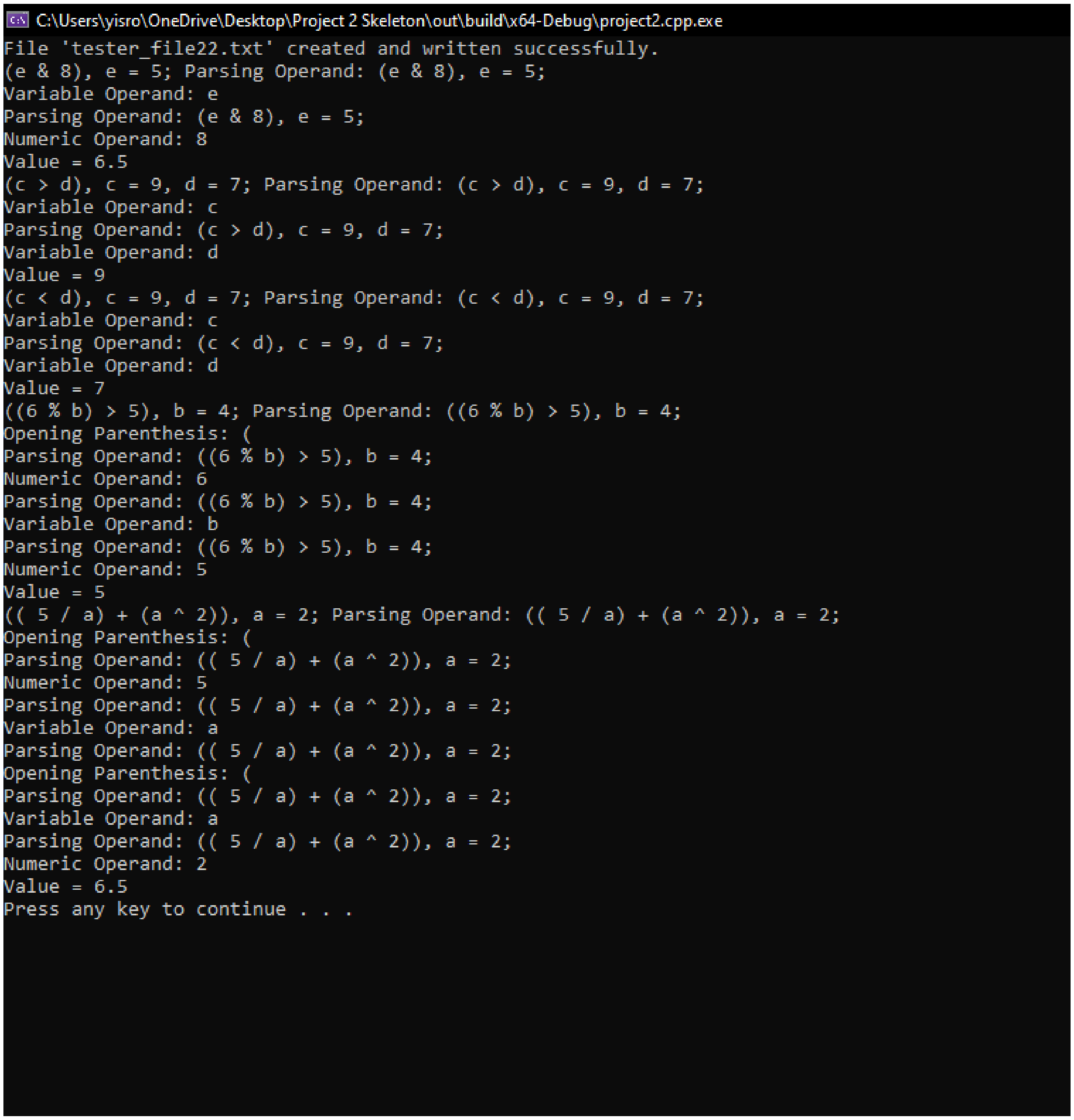
Discussion of approach included
I faced a lot of new challenges during this project. While the basic building blocks of how to set up the grammar and how to write code in general was familiar to me, all the specifics of C++ was not as familiar to me than, let's say, javascript or Java, or even python. I know those languages better. That said I was familiar enough with C++ to build out the skeleton code that came with the project. And with some simple googling about specific syntax, I was able to write out the functionality for the various new operations. I took this project one operation at a time. I started with the minus, then multiplication, then exponents, etc… Doing like this made it manageable and I got quicker with setting up each operation as I went along. I tested them with the example provided in the project description and they worked like a charm.

Lessons learned included
As I mentioned, I learned more about c++ during this project. It was a learning experience that exposed me to file structure of c++, which differs from Java or Javascript in many ways. However, I was pleasantly surprised by how similar coding languages are in terms of writing code. The variable instantation is similar, the class definitions are similar. Of course they have differences but the basic idea is the same.

I also learned more about how to parse a string. The format that the expression are given is the expression itself, then what the values of each variable is. It is written in a way where order of operations is in place with the parentheses. And this structure is what the code is expecting to see. It searches for open and close parentheses to split up the operations.

Set up
The input file is read from the folder Project 2 yisroel arnson\out\build\x64-Debug, which is 3 folders down from the code files. This is where it is able to be read from.

Test Cases
The following screenshot is for the following test cases.



```
C:\Users\yisro\OneDrive\Desktop\Project 2 Skeleton\out\build\x64-Debug\project2.cpp.exe
File 'tester_file22.txt' created and written successfully.
(e & 8), e = 5; Parsing Operand: (e & 8), e = 5;
Variable Operand: e
Parsing Operand: (e & 8), e = 5;
Numeric Operand: 8
Value = 6.5
(c > d), c = 9, d = 7; Parsing Operand: (c > d), c = 9, d = 7;
Variable Operand: c
Parsing Operand: (c > d), c = 9, d = 7;
Variable Operand: d
Value = 9
(c < d), c = 9, d = 7; Parsing Operand: (c < d), c = 9, d = 7;
Variable Operand: c
Parsing Operand: (c < d), c = 9, d = 7;
Variable Operand: d
Value = 7
((6 % b) > 5), b = 4; Parsing Operand: ((6 % b) > 5), b = 4;
Opening Parenthesis: (
Parsing Operand: ((6 % b) > 5), b = 4;
Numeric Operand: 6
Parsing Operand: ((6 % b) > 5), b = 4;
Variable Operand: b
Parsing Operand: ((6 % b) > 5), b = 4;
Numeric Operand: 5
Value = 5
(( 5 / a) + (a ^ 2)), a = 2; Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Opening Parenthesis: (
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Numeric Operand: 5
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Variable Operand: a
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Opening Parenthesis: (
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Variable Operand: a
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Numeric Operand: 2
Value = 6.5
Press any key to continue . . .
```

#1

(e & 8), e = 5;

When given this expression, the program returns the following

(e & 8), e = 5; Parsing Operand: (e & 8), e = 5;
Variable Operand: e
Parsing Operand: (e & 8), e = 5;

```
Numeric Operand: 8
Value = 6.5
```

6.5 is the average of 5 and 8! So this passes.

#2

(c > d), c = 9, d = 7;

When given this expression, the program returns the following

```
(c > d), c = 9, d = 7; Parsing Operand: (c > d), c = 9, d = 7;
Variable Operand: c
Parsing Operand: (c > d), c = 9, d = 7;
Variable Operand: d
Value = 9
```

The program is supposed to decide which value is greater, c or d. It returns the larger value (9) successfully.

#3

(c < d), c = 9, d = 7;

When flipped around to search for the smaller number, the program also successfully does so

```
(c < d), c = 9, d = 7; Parsing Operand: (c < d), c = 9, d = 7;
Variable Operand: c
Parsing Operand: (c < d), c = 9, d = 7;
Variable Operand: d
Value = 7
```

#4

((6 % b) > 5), b = 4;

The program prints out the following response for this expression

```
((6 % b) > 5), b = 4; Parsing Operand: ((6 % b) > 5), b = 4;
Opening Parenthesis: (
Parsing Operand: ((6 % b) > 5), b = 4;
Numeric Operand: 6
Parsing Operand: ((6 % b) > 5), b = 4;
```

```
Variable Operand: b
Parsing Operand: ((6 % b) > 5), b = 4;
Numeric Operand: 5
Value = 5
```

This is correct because it is finding 6 mod 4, which goes in once with a remainder of 2. Then it compares if 5 or 2 is greater, and it returns 5 correctly.

#5

((5 / a) + (a ^ 2)), a = 2;

This expression contains two sets of terms. It successfully returns 6.5

```
(( 5 / a) + (a ^ 2)), a = 2; Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Opening Parenthesis: (
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Numeric Operand: 5
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Variable Operand: a
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Opening Parenthesis: (
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Variable Operand: a
Parsing Operand: (( 5 / a) + (a ^ 2)), a = 2;
Numeric Operand: 2
Value = 6.5
```